CSC 212 Midterm Study Guide

1. C++ Review, Memory, Pointers

In C++, memory is managed using stack and heap. Stack memory is used for local variables, while heap memory is dynamically allocated using pointers.

Common pointer operations:

- int* ptr = &var; // Pointer to a variable
- *ptr = 10; // Dereferencing a pointer
- ptr = new int[10]; // Dynamic array allocation
- delete[] ptr; // Freeing memory

Avoid dangling pointers by properly deallocating memory.

2. Introduction to Analysis of Algorithms

Algorithm analysis helps in determining the efficiency of an algorithm. We use asymptotic notation:

- O(f(n)): Upper bound (Worst case)
- Omega(f(n)): Lower bound (Best case)
- Theta(f(n)): Tight bound (Average case)

Example: Analyzing a loop iterating from 1 to n gives O(n) complexity.

3. Big-O Notation (Detailed)

Big-O	Category	Example
O(1)	Constant Time	Accessing an array element
O(log n)	Logarithmic Time	Binary search
O(n)	Linear Time	Looping through an array
O(n log n)	Log-Linear Time	Merge Sort
O(n^2)	Quadratic Time	Nested loops (Bubble Sort)

4. Dynamic Arrays

Dynamic arrays grow when needed by doubling in size. This allows O(1) average-time insertions but requires memory reallocations.

C++ Example:

```
vector<int> arr; // Dynamic array using std::vector
arr.push_back(5); // Append element
```

5. Stacks

Stacks follow Last In First Out (LIFO) principle. Operations include:

- push(): Add element
- pop(): Remove top element
- peek(): View top element

Example:

```
stack<int> s;
s.push(10);
s.pop();
```

6. Practice Problems

1. What is the Big-O complexity of a function with a single loop running from 1 to n?

2. Given a nested loop iterating from 1 to n in both inner and outer loops, what is its complexity?

3. Write a C++ function to implement a stack using an array.

4. If an array doubles in size each time it reaches capacity, what is the amortized cost of insertions?

5. Compute the time complexity of QuickSort in best and worst case scenarios.

7. Answer Key

1. O(n) - The loop runs 'n' times.

2. O(n^2) - Nested loops result in quadratic complexity.

3. A stack can be implemented using an array and a top pointer.

4. O(1) - The amortized insertion cost remains constant due to exponential resizing.

5. Best-case: O(n log n) (even splits); Worst-case: O(n^2) (unbalanced splits).